

CSI 333 – Programming at the Hardware-Software Interface

Some Examples of C Programs (continued)

Handout 4.1

Program Example 7: This example contains a collection of functions that use a structure and an array of that structure. (This is *not* a complete program.)

```
#include <stdio.h>
#include <string.h>

/* Max. number of characters in an employee's name. */
#define MAX_NAME_LEN 25

/* Maximum number of employees in an organization. */
#define MAX_EMPLOYEES 1000

/* Type definition for the employee structure (record). */
struct employee {
    char name[MAX_NAME_LEN]; int age; float salary;
};

/* Use of typedef to set up a synonym for the above struct. */
typedef struct employee EmpRec;

/* Prototypes for all the functions shown in this handout. */
void print_record (EmpRec);
void employee_search (EmpRec [], int, char []);
void raise_salary (EmpRec [], int, float);

/*-----*/
/* Code for the print_record function. */
/*-----*/

void print_record (EmpRec x) {

    /* The input parameter x represents an employee record. This
       function simply prints to stdout the various fields of the record.
    */

    printf("Employee's name   = %s\n", x.name);
    printf("Employee's age     = %d\n", x.age);
    printf("Employee's salary = %f\n", x.salary);

} /* End of print_record */
```

```

/*-----*/
/* Code for the employee_search function. */
/*-----*/

void employee_search (EmpRec division[], int num_emp, char emp_name[]) {

    /* The inputs to this function are:

        division[] -- The array of employee records for the division.

        num_emp    -- No. of employees in the division. (It is assumed
                     that division[0] through division[num_emp -1]
                     contain the records of the employees.

        emp_name   -- Name of the employee whose information is to
                     be printed.

    The function searches the records (sequentially) for the record
    of the employee whose name matches the one given by emp_name.
    If such a record is found, all the information about that employee
    is printed. Otherwise, an error message is produced.
    */

    int i;    /* Temporary; loop index. */

    /* Search the array of records. Use the strcmp function to compare */
    /* employee's names with the emp_name parameter. */

    for (i = 0; i < num_emp; i++) {

        if (strcmp(division[i].name, emp_name) == 0) {

            /* Found a record where the name matches emp_name. */
            print_record(division[i]); return;

        }

    } /* End of for loop. */

    /* If control reaches this point, then there was no match. */
    printf("There is no employee with name: %s\n", emp_name);

} /* End of employee_search */

```

```

/*-----*/
/* Code for the raise_salary function. */
/*-----*/

void raise_salary (EmpRec division[], int num_emp, float percent_raise) {

    /* The inputs to this function are:

        division[] -- The array of employee records for the division.

        num_emp    -- No. of employees in the division. (It is assumed
                     that division[0] through division[num_emp -1]
                     contain the records of the employees.

        percent_raise -- The percentage raise (e.g. if this value is 5.0,
                          then the employee's new salary becomes 1.05 times
                          the current salary).

        Since arrays are passed by reference in C, this function can
        directly change the salary field of each employee record.
    */

    int i;    /* Temporary; loop index. */
    float raise_factor; /* The factor to compute new salary. */

    /* Compute the value of raise_factor. */
    raise_factor = 1.0 + percent_raise/100.0;

    for (i = 0; i < num_emp; i++) {

        /* Raises each employee's salary using raise_factor. */
        division[i].salary *= raise_factor;

    } /* End of for loop. */

} /* End of raise_salary. */

```